Chapter 2
Introduction to Computational Techniques

Computational techniques are fast, easier, reliable and efficient way or method for solving mathematical, scientific, engineering, geometrical, geographical and statistical problems via the aid of computers. Hence, the processes of resolving problems in computational technique are most time step-wise. The step-wise procedure may entail the use of iterative, looping, stereotyped or modified processes which are incomparably less stressful than solving problems-manually. Sometimes, computational techniques may also focus on resolving computation challenges or issues through the use of algorithm, codes or command-line. Computational technique may contain several parameters or variables that characterize the system or model being studied. The inter-dependency of the variables is tested with the system in form of simulation or animation to observe how the changes in one or more parameters affect the outcomes. The results of the simulations, animation or arrays of numbers are used to make predictions about what will happen in the real system that is being studied in response to changing conditions.

Due to the adoption of computers into everyday task, computational techniques are redefined in various disciplines to accommodate specific challenges and how they can be resolved. Fortunately, computational technique encourages multi-tasking and interdisciplinary research. Since computational technique is used to study a wide range of complex systems, its importance in environmental disciplines is to aid the interpretation of field measurements with the main focus of protecting life, property, and crops. Also, power-generating companies that rely on solar, wind or hydro sources make use of computational techniques to optimize energy production when extreme climate shifts are expected. In this case, engineers, scientists and environmentalist are combining computational and meteorological dataset to address the challenge of understanding, characterizing, and predicting complex environmental systems. The most difficult task in computational techniques is understanding the computer programming language. A programming language is a formal language that highlights sets of instructions for execution. programming language is grouped by types namely Array languages, Assembly languages, Authoring languages, Constraint programming languages, Command line interface languages, Compiled lan-

M. E. Emetere, *Environmental Modeling Using Satellite Imaging and Dataset Re-processing*, Studies in Big Data 54,
https://doi.org/10.1007/978-3-030-13405-1_2

guages, Concurrent languages, Curly-bracket languages, Dataflow languages, Data-oriented languages, Decision table languages, Declarative languages, and Embeddable languages. Array language is programming language that is used to convert operations from scalars to vectors, matrices, and higher-dimensional arrays. A typical example of modern languages that supports array language includes the following: Fortran 90, Mata, MATLAB, Analytica, TK Solver (as lists), Octave, R, Cilk Plus, Julia, Perl Data Language (PDL) and the NumPy extension to Python. An assembly language is a group of low-level programming languages used by microprocessors and other programmable devices to implement symbolic representation of machine code needed to program a given CPU architecture. The type of assembly language includes: Complex Instruction-Set Computer (CISC), Reduced Instruction-Set Computer (RISC), Digital Signal Processor (DSP) and Very Long Instruction Word (VLIW).

Authoring language is a notation used to control the appearance and functionality of webpages when displayed in a browser. Example of authoring language are DocBook, DITA, PILOT, TUTOR, Bigwig, Chamilo, Hollywood (Hollywood Designer graphical interface), Learning management system, SCORM, Web design program, XML editor and Game engine. Constraint programming is a type of programming paradigm that displays variable in the form of constraints. Libraries that accepts constraint programming are Artelys Kalis, C++, Java, Python library, FICO Xpress module, Cassowary, Smalltalk, Ruby library (LGPL), CHIP V5, Choco, Java library, Comet, Cream, Java library (LGPL), Disolver, Facile, OCaml library (CC0 1.0), finite-domain, Haskell library (MIT), Gecode, Google OR-Tools, JaCoP, Java library, LINDO MonadicCP, Haskell library (BSD-3-Clause) etc. Command line interface languages are languages that are used to interact with a computer program where the user issues commands to the program in the form of successive lines of text. The successive lines of text are called command lines. The command lines are executed in the shell of operating system. The shell of operating system includes AmigaOS (Amiga CLI/Amiga Shell), Unix OS (Bourne shell, Almquist shell, Debian Almquist shell).

Bash, Korn shell, Z shell, C shell, TENEX C, Emacs shell, rc shell rc, Standalone shell and Remote shell), Microsoft Windows (CMD.EXE, Windows PowerShell, Hamilton C shell, 4NT, Recovery Console), DOS(COMMAND.COM, 4DOS, NDOS and GW-BASIC), OS/2 (CMD.EXE, Hamilton C, and 4OS2), IBM OS/400 (AS/400 Control Language, iSeries QSHELL) Apple (Apple DOS/Apple ProDOS) and Mobile devices (DROS, Java ME platform).

Compiled languages is a programming language whose implementations are typically compilers (translators that generate machine code from source code), and not interpreters (step-by-step executors of source code). Examples of compiled programming language are ALGOL, BASIC, C, D, CLEO, COBOL, Cobra, Crystal, Eiffel, Fortran, Go, Haskell, Haxe, JOVIAL, Julia, LabVIEW, G, Pascal, SPITBOL, Visual Foxpro and Visual Prolog. Embeddable languages are programming language that supports scripting in real-time systems. Example of embedded language includes Python, C++ and Java.

The utmost goal of researchers or modeler is to develop an experimentally driven computer model that generates accurate predictions of environmental scenarios e.g. Numerical Weather Prediction (NWP)model. A typical example of an experimentally driven computer model is the Numerical Weather Prediction (NWP) developed by the National Oceanic and Atmospheric Administration (NOAA). NWP is a form of day-to-day weather model data. NWP focuses on taking current observations of weather and processing these data with numerical computer models to forecast the future state of weather. In this case, current weather or meteorological observations serve as input to the numerical computer models through a process known as data assimilation to produce outputs of temperature, precipitation, and hundreds of other meteorological elements from the oceans to the top of the atmosphere (NOAA 2018). The NWP data are Global Ensemble Forecast System (GEFS), Global Forecast System (GFS), Climate Forecast System (CFS), North American Multi-Model Ensemble (NMME), North American Mesoscale (NAM), Rapid Refresh (RAP) and Navy Operational Global Atmospheric Prediction System (NOGAPS). The assimilation data of the NWP is obtained from the Global Data Assimilation System (GDAS). Before the usage of assimilation data in environmental model, optimum interpolation (OI) method was used. The OI was introduced by L. S. Gandin (István et al. 2013). Its use was discontinued because of the limitation/drawbacks. The variational method replaced the OI until it was also discontinued mainly because of some programing and computational difficulties it encountered.

In the early days of understanding the role of computers in fostering research, some profession e.g. management, economics, biology etc. makes use of statistical packages only. In recent time, advancement of computers and computer application has brought about more sophisticated computer packages for solving problems (Emetere and Sanni 2015). In this chapter, we shall discuss on general outline on computational techniques, open-software packages and libraries. In science and engineering, computational technique is beyond using computer application. It entails restructuring related mathematical or physical theories to solve or optimize a specific process.

2.1 General Outline of Computational Techniques

What generally comes to mind when computational technique is mentioned is computer software. Computer software are structured algorithms or codes that implement a specific task. Hence, the three broad classification of computer software namely system software, application software and programming software. System software are made-up of operating system (Microsoft Windows, Mac OS X, and Linux) device drivers (Bios, motherboard drivers, hardware divers, virtual device drivers, sound cards etc.), servers and software components. Application software (AS) is used for attaining specific tasks. The types of AS available in the market includes licensed, sold, freeware, shareware, and open source software. In this chapter, the emphasis shall be on the open source (since most environmental models are open source).

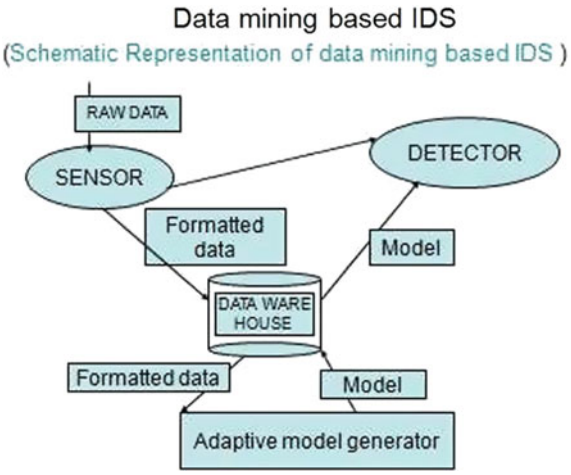

Fig. 2.1 Computational technique for data mining (Chandra and Chinmayee 2012)

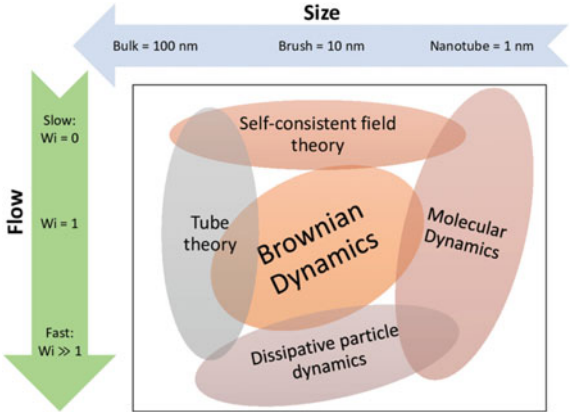

Fig. 2.2 Computational technique for materials (Korolkovas 2016)

Programming software are mini-codes or macros used for writing programs through tools such as editors, linkers, debuggers, compilers/interpreters etc.

Computational techniques may be seen as a very broad concept in modern research. The early type of computational technique is termed computer aided algebraic systems. In this technique, algebraic related problems are solved using algorithms, codes, charts and special syntaxes. However, as research expanded, there were need for more application software for special task. In environmental modelling, there are general and specific application software. The general application software performs tasks as speculated by the modeler discretion. Example of general application software in environmental modelling includes MATLAB, MATHCAD, Origin, Homer, GNU plot, FreeMat, Octave, Microsoft Excel, Magma, Maple, R package, Mathematical, SageMath, PolyMath, SMath, COMSOL etc. The various computational techniques are shown pictorially in Figs. 2.1 and 2.2.

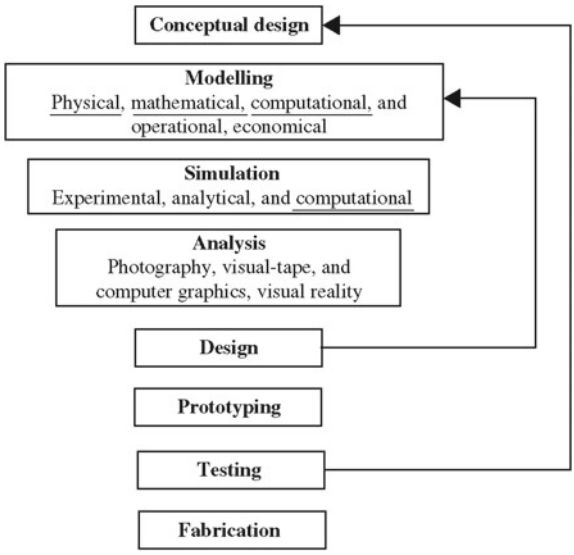

Fig. 2.3 Pictorial illustration of computational technique

The specific application software (Captera 2017) for environmental modelling includes SimaPro or GaBi (for assessing Carbon Footprints), Qual2K (river and stream water quality and process interaction model), Hec-HMS (for hydrologic modeling system), WaterCAD (for designing and planning hydrologic systems), ADORA or AERMOD (for monitoring air pollution), EMEX (for managing incidents and track corrective and preventative actions), Enviro (for gathering, managing, and displaying lab and field data for water, soil, and air), SmartData (for investigating environmentally contaminated sites), Accuvio (for monitoring carbon print and greenhouse gases) etc. In recent time, it has been proven that some environmental application software has their flaws (Patwardhan 2016). This development is quite worrisome because beginners and young professionals in environmental studies may be building on faulty foundation.

Based on the aforementioned, environmental scientists are advised to use either the general application software or adopt open-source application/library to prevent uncontrollable error-prone analysis/research. The outline of the computational technique will be explained using the pictorial illustration in Figs. 2.1, 2.2 and 2.3.

The fabrication stage is referred as the developmental procedures where the modeler or researcher decides on what kind of computational technique would be appropriate for the model. Most modeler at this stage decides to adopt mathematical methods. Mathematical method is a tool for solving abstract and real problems. Mathematical method is fondly used in science, engineering, social science and humanities. The list of method that are usually considered in mathematical methods includes: Infinite series, power series, Complex numbers, Integral transform, Wavelet transform, Fourier transform spectroscopy, Harmonic analysis, Linear algebra, Partial differentiation, Multiple integrals, Vector analysis, Fourier series and

transforms, Ordinary differential equations, Calculus of variations, Linear function, Quadratic function, Cubic function, Quartic function, Cabibbo-Kobayashi-Maskawa matrix, Density matrix, Fundamental matrix, Fuzzy associative matrix, Gamma matrices, Gell-Mann, Hamiltonian matrix, Wall polynomial, Wangerein functions, Weber function, Weierstrass function, Weisner's method, Whittaker function, Wilson polynomial, Irregular matrix, Overlap matrix, S-matrix, State transition matrix, Substitution matrix, Z-matrix, Quintic function, Sextic function, Tensor analysis, Special functions, Schubert polynomial, Schur polynomial, Selberg integral, Sheffer polynomial, Slater's identities, Stieltjes polynomial, Stieltjes–Wigert polynomials, Strömgren function, Struve function, Legendre function, Bessel function, Hermite function, Laguerre function, Partial differential equations, Functions of a complex variable, Integral transforms, Gamma function, Barnes G-function, Beta function, Digamma function, Polygamma function, Incomplete beta function, Incomplete gamma function, K-function, Multivariate gamma function, Student's t-distribution, Probability and statistics, Two-sided Laplace transform, Mellin transform, Laplace transform, Fourier transform, Fourier series, Sine and cosine transforms, Hartley transform, Short-time Fourier transform, Celine's polynomial, Charlier polynomial, Pafnuty Chebyshev, Chebyshev polynomials, Painlevé function, Painlevé transcendents, Poisson–Charlier polynomial, Pollaczek polynomial, Cyclotomic polynomials, Rectangular mask short-time Fourier transform, Gegenbauer polynomials, Gottlieb polynomial, Gould polynomial, Gudermannian function, Chirplet transform, Fractional Fourier transform (FRFT), Hankel transform, Hall polynomial, Hall–Littlewood polynomial, Hankel function, Heine functions, Racah polynomial, Riccati–Bessel function, Riemann, zeta-function, Rodrigues formula, Rogers–Askey–Ismail polynomial, Rogers–Ramanujan identity, Rogers–Szegő polynomials, Hermite polynomials, Heun's equation, Horn hypergeometric series, Hurwitz zeta-function, Boubaker polynomial etc. However, some modeler would want to generate their own brand of mathematical methods.

Secondly, the modeler choose what kind of programming language would be appropriate to solve the mathematical method that has been adopted. The common programming language used in research in modern times include: Python, Q# (Microsoft programming language), C, C–, Java, C++, C#, MATLAB, Mathcad, Visual Fortran.

Visual FoxPro, JavaScript, JCL, Jython, MATH-MATIC, Visual J++, Visual J#, Mathematica etc. Few researcher or modeler used one or more of the aforementioned programming language.

As shown in Fig. 2.3, it is mandatory for the researcher or modeler computationally validate the solved problem using dataset. If the modeler is not satisfied with the outcome, he checks his computational procedures once more until he/she can figure-out how to obtain an accurate outcome. Once the modeler is satisfied with the testing of the computational work, he moves on to the prototyping of the computational work. Most modeler stops at the testing stages because of funds or technical know-how. The prototype is sent to research centers or industries for feedback purposes.

Once the feedback comes out from the sources, the modeler goes on to the designing stage. At this point the computational work comes out as a licensed or open-source application. The modelling, analysis and simulation stages are performed by the end-users.

2.2 Open Source Scientific Packages

The open source scientific packages are generally referred to as computer software that are licensed either under the free software licenses or the open-source licenses. However, the term 'open-source' may not necessary be without financial involvement as there are commercial open-source applications that are linked to business models e.g. AdaControl, Sun Studio, Dolibarr, Openbravo, EyeOS, Kaltura, LucidWorks, Zenoss Core, OrangeHRM, Qt, Talend Open Profiler etc. The idea of the open source can be illustrated as shown in Fig. 2.4. The most important factor of the open source is the possibility of low transaction costs which may only occur if the supportive library of the open source application is not available on the local computer.

There are notable scientific open source software applications that have gained relevance in its application in environmental studies. Tabula is a scientific package that allows users to extract data from pdf into a CSV spreadsheet using a simple and easy-to-use interface. This package is particularly useful when there is large data in the pdf format. However, the shortcoming of this software is that the processing speed is slow. Dakota is another freely available engineering software framework for large-scale optimization and uncertainty analysis. Dakota software's advanced

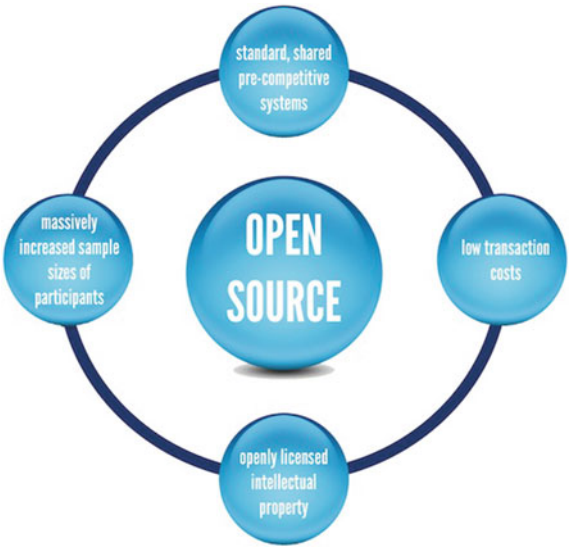

Fig. 2.4 The ideals of open source (Wilbanks 2013)

parametric analyses enable design exploration, model calibration, risk analysis, quantification of margins and uncertainty with computational models. This features are very important in environmental engineering because it helps to improve on the accuracy of predictions. The summarized list of scientific software includes, 3D Slicer, AMPHORA2, Ascalaph Designer, Bioconductor, BioModels Database, Biskit, Brian Simulator, ChemTool, Cn3D, DataMelt, EMBOSS, Emergent, GenMAPP, GENtle, GIMIAS, Gnaural, Gwyddion, HMMER, ImageJ, InVesalius 3, LabKey, MDynaMix, OpenMS, OsiriX Imaging Software, PathVisio, QuteMole, RasMol and OpenRasMole, RDQA, Spatiotemporal Epidemiological Modeler (STEM), UGENE, Virtual Cell (VCell), XDrawChem, ZygoteBody etc.

In recent times, researchers or modelers seek to build their own open source application. Ibrahim (2010) in his blog simplified the foundational requirements to begin open source project with six salient questions i.e.

i. Can we financially sponsor the project? Do we have an internal executive champion?
ii. Is it possible to join efforts with an existing open source project?
iii. Can we launch and maintain the project using the OSS model?
iv. What constitutes success? How do we measure it?
v. Will the project be able to attract outside enterprise participation (from the start)?
vi. Is there enough external interest to form and grow a developer community?

The key to open-source development is the ability of the researcher to understand the licenses of the programs he/she will be using to create his/her open source application. Compatibility issues between licenses can originate when you are trying to include open source code as a library in your existing project. Hence, it is very important for a modeler to understand in clear terms the details of each licenses to avoid copyright infringement. For example, Seher (2017) explained that software package licensed under an Apache 2.0 license are compatible with software license of GNU General Public License, version 3 (GPLv3) because Apache 2.0 terms are covered by GPLv3 because both licenses have same patent usage protection. The GPLv3 license reads: the source code must be made public whenever a distribution of the software is made; modifications of the software must be released under the same license; changes made to the source code must be documented; If patented material was used in the creation of the software, it grants the right for users to use it. If the user sues anyone over the use of the patented material, they lose the right to use the software. However, GPLv2 license is not compatible with Apache 2.0 because of patent grant clause that is missing. License compatibility can therefore be considered as a subtle danger that must be considered carefully. Some authors have given insight on how to avoid copyright infringement. For example, Välimäki (2005) reported a schematic sketch of license compatibility as presented in Fig. 2.5.

The term derivative work in Fig. 2.5 refers to as everything that uses the source code in any way possible. For example, Berkeley Software Distribution (BSD) 2-clause supports derivative work. However, 3-clause of BSD do not support derivative work because it states "*the names of the author and contributors can't be used to promote products derived from the software without permission*". This gives rise

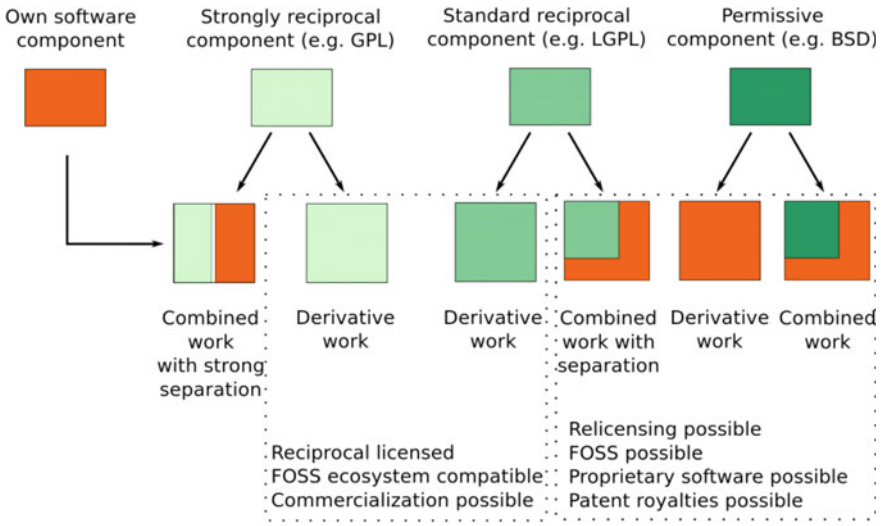

Fig. 2.5 License compatibility issues (Välimäki 2005)

to the question: "what is open source license". Readers should take a little time to go through the definition of open source by Open Source Initiative (OSI 2018). The four component of the open source licenses are: software can be modified, used commercially, and distributed; software can be modified and used in private; a license and copyright notice must be included in the software; software authors provide no warranty with the software and are not liable for anything.

The second key to open-source development is the organization of the new project. Flory (2018) reported in his article that managing an open source project is a challenging work, and the challenges grow as the project grows because at every developmental stages of the project, it is expected that the project meet different requirements and span multiple repositories. The author gave three tips for organizing an open source project. The three tips are: bring development discussion to issues and pull requests (sincerity and transparency is advised); Set up kanban-style project boards (projects boards are repository boards that are used in a single repository) and Organization (boards for use in a GitHub organization across multiple repositories); Build project boards into your workflow. The workflow is pictorially defined in the chart presented in Fig. 2.6. For in-depth reading, readers are advised to go to the link presented in the reference section.

The third key to open-source development is the strict adherence of author or modeler to maintain openness. This process helps expert to modify features in the project. Sometimes, developers express themselves quite bluntly, so the modeler must be emotionally balance to take the message and discard the insults. Also, inexperience volunteers may cause set-backs in the open source development. In this case, suggestions or contributions from experts are out rightly not in line of discuss.

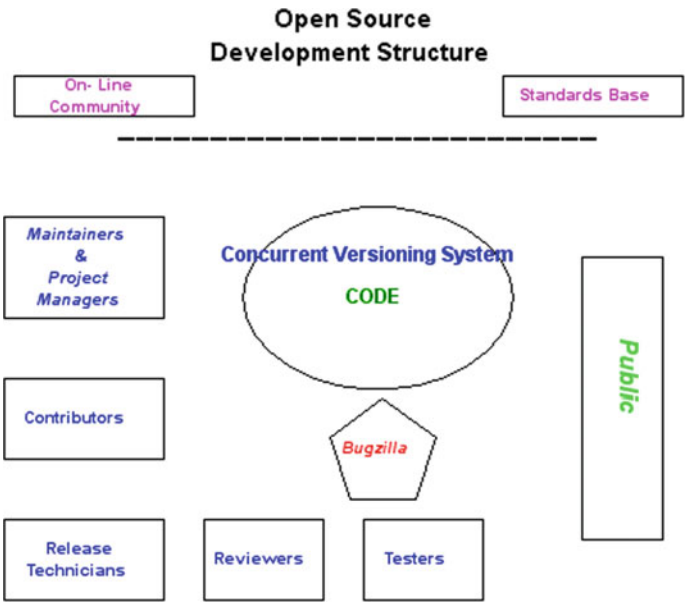

Fig. 2.6 Open source development structure (Tom 2004)

Fig. 2.7 Single vendor open source projects (Matthias 2013)

The ability of the modeler to decipher between useful and wasteful contributions is very important for the onward progress of the open source project.

Matthias (2013) stated clearly that open source software is not only about programming code but it is a carefully organized process that is hinged on a systematic order. In the light of his explanation, Matthias (2013) identified four main organizations within the open source community namely: single vendor open source projects; development communities; user communities; and open source competence centers. The pictorial chart that explicitly describes the mode of operations is presented in Figs. 2.7, 2.8, 2.9 and 2.10.

In environmental research, there are about a thousand open-source scientific packages that are used currently in the field. NASA (2018) listed several open-source that can be operated by novice and professionals. Most professional open-source appli-

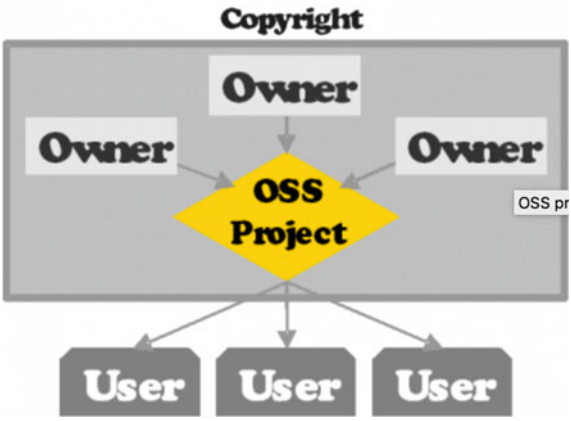

Fig. 2.8 Development communities (Matthias 2013)

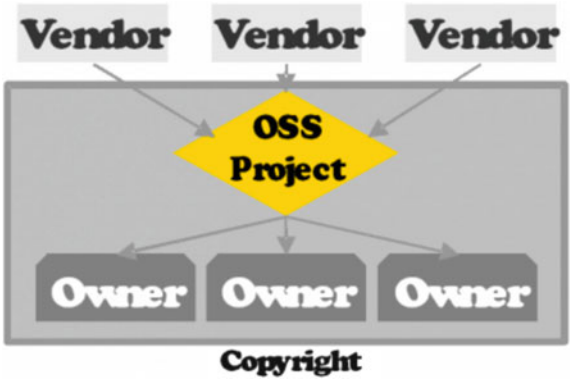

Fig. 2.9 User communities (Matthias 2013)

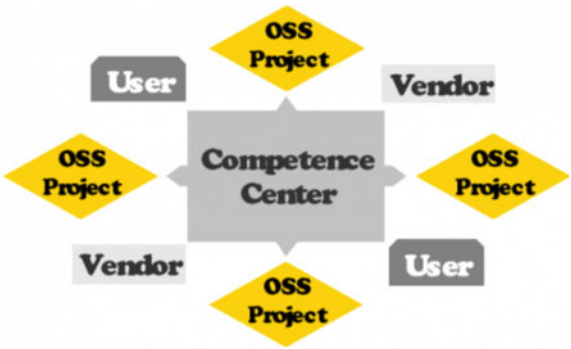

Fig. 2.10 Open source competence centers (Matthias 2013)

cation packages are written in Python or C++ programming language. One of the many reasons for adopting Python or C++ in open-source application is because of its flexibility to accommodate the 'big data' concept. Data science space adopts the C++ language because of the nature of its operation.

Open-source applications that are written in Python, Java or C++ language are adjudged appropriate: when complex machine learning algorithms are involved; when the dataset is in terabyte or petabyte; when working on deep learning and deep neural networks. However, researchers have noted that few open-source application packages are obsolete. Upasani (2016) noted that open-source application/libraries need to embrace digital technologies and library management systems (LMS) in order to work smart and achieve more with less. Higgs (2016) believes that open source software is unsupported, unsustainable and unreliable. Hence, big companies do not patronize open-source software. ConnectUS (2018) also highlighted the disadvantages of open-source software namely vulnerability to malicious users, not user-friendly as commercial versions and do not come with extensive support.

The preferred open-source application used by the author is the CERN-Root software. ROOT is an object-oriented program and library developed by European Organization for Nuclear Research (CERN). It was originally designed for particle physics data analysis but it is also used in other applications such as astronomy and data mining. Root are used for plotting histograms and graphs, curve fitting, statistical analysis, data analysis, matrix algebra, four-vector computations, multivariate data analysis, image manipulation, 3D visualizations (geometry), creating files in various graphics formats, interfacing Python and Ruby code in both directions and interfacing Monte Carlo event generators.

2.3 Open-Source Library

Library is a collection of non-volatile resources used by computer programs, often to develop software. Libraries are used for software development to enhance the software to perform specific task. The major computer libraries are written in terms of language e.g. Multi-language, C, C++, Delphi, .NET Framework languages (C#, F#, VB.NET and PowerShell), Fortran, Java, Scala, Perl, Python, Groovy, XNUMBERS, INTLAB etc.

Open-source library has almost the same shortcoming as open-source application. Since most open-source allows anyone to interact with its source codes, the open-source library can be modified to suite any task. For example, Igor (2017) modified added commits, contributors count and other metrics from Github to enhance the proxy metrics for Python library popularity.

Open-Source library are designed to perform certain function. Also, library can be built upon one another. For example, NumPy is a Python library that provides a fundamental framework where scientific computation stack is built. The main functionality of SciPy library is built upon NumPy. Most open-source libraries are low-level tool i.e. it requires more codes by the modeler to advance its status to a

high-level tool. For example, the matplotlib library cannot exist alone, except it is enhanced by the stack of NumPy, SciPy and Pandas Python library.

One of the preferred open-source library-used by the author is the OpenCV. OpenCV is an open and free source computer vision library that is released under a BSD license. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV is used for real-time applications like image processing, matrix algebra, four-vector computations, multivariate data analysis, image manipulation, 3D visualizations (geometry), creating files in various graphics formats, interactive art, mines inspection evaluation, stitching maps on the web or through advanced robotics. OpenCV is enabled with OpenCL to boost its hardware acceleration of the underlying heterogeneous computational platform.

The C++ libraries includes: Boost, GSL, BDE, Dlib, JUCE, Loki, Reason, yomm2, Folly, Abseil, cxxomfort, libsourcey, OnPosix, Ultimate++, CAF, cpp-mmf, CommonPP, Better Enums, Smart Enum, nytl, SaferCPlusPlus, fcppt, bit-field.h, composite_op.h, Yato, Kangaru, yaal, rpnx-serial,libnavajo, C++ RESTful framework, C++ REST SDK, cpr, cpp-netlib, cpp-redis, tacopie, Boost.Asio, Boost.Beast, gsoap,POCO, omniORB, ACE, TAO, wvstreams, Unicomm, restful_mapper, zeromq, curlpp, Apache Thrift, libashttp, Simple C++ REST library, libtins, PcapPlusPlus, HTTPP, The Silicon C++14 Web Framework, ngrest, restc-cpp, OpenDDS, Breep, uvw, rest_rpc, EasyHttp, nghttp2,Dear ImGui, FLTK,nana[, WxWidgets, OWLNext,tiny file dialogs, Switch, glibmm, gtkmm, goocanvasmm, libglademm, libgnomecanvasmm, webkitgtk, flowcanvas, evince, Qt, qwtplot3d, qwt5, libdbusmenu-qt, QuickQanava, QuickProperties, SFML (Simple and Fast Multimedia Library),SDL (Simple DirectMedia Layer), SIGIL (Sound, Input, and Graphics Integration Library), Cinder, openFrameworks, cairomm, nux, pangomm, gegl, stb, Adobe/boost GIL, GraphicsMagick, Skia Graphics Engine, enwiki:Skia_Graphics_Engine, Anti-Grain Evolution, plotutils, libraw, openexr, qimageblitz, imagemagick, djvulibre, poppler, SVG++, id3lib, taglib, opencv, dlib, ITK, OTB, Vulkan, OpenGL, bgfx, Ogre3D, Diligent Engine, GLEW, GLAD, Epoxy, GLFW, GLM, hlsl++, assimp, VTK, Magnum, Irrlicht, Horde3D, Visionaray, Open CASCADE, OpenSceneGraph, EntityX, Anax, EntityPlus, EnTT, BOX2D, stats++, StatsLib, alglib, ArrayFire High Performance Computation Library, GNU MP bignum C++ interface, BigNumber, Boost.Multiprecision, Boost.Math.Special Functions and Statisticalistributions, Boost.Random, NTL - A Library for doing Number Theory, cpp-measures, G + Smo cross-platform library for isogeometric analysis, Exact floating-point arithmetic library, Boost.uBLAS, Eigen, Armadillo, Blitz++, IT++, Dlib - linear algebra tools, Blaze, ETL, DecompLib, OptimLib, Boost.Graph, LEMON, OGDF—Open Graph Drawing Framework, NGraph—a simple (Network) Graph library in C++, GTpo, cln, Dlib—machine learning tools, MLPACK—machine learning package, Shogun—large scale machine learning toolbox, CGAL—Computational geometry algorithms library, Wykobi—Computational geometry library, PCL—Point Cloud library, yasmine—C++11 UML state machine framework, libxml++, pugixml, tinyxml, tinyxml2, Xerces, gSOAP, ai-xml, json, ArduinoJson, jsonme–, ThorsSerializer (JSON/YAML Input Output Streams), JsonBox, jsoncpp, zoolib, JOST, CAJUN, libjson, nosjob, rapidjson, jsoncons, JSON++,

qjson, json-cpp, jansson, json11, JSON Voorhees, jeayeson, ujson, minijson, jios (JSON Input Output Streams), Botan, gnutls, openssl, crypto++, TomCrypt etc.

The Python libraries includes: ADOdb, AppJar, Beautiful Soup (HTML parser), CGAL, CheetahTemplate, Construct (python library), Cubes (OLAP server), Genshi (templating language), Gensim, IronPython, Jinja (tmplate engine), Kamaelia, Kid (templating language), Kivy (framework), Natural Language Toolkit, Pickle (Python), PLWM, PyEphem, Pygame, Pyglet, PyGObject, PyGTK, PyObjC, PyQt, PySide, Python Imaging Library, Python Robotics, Python-Ogre, RDFLib, Redland RDF Application Framework, Requests (software), RPyC, SimpleITK, SimPy, Sound Object (SndObj) Library, Soya3D, SpaCy, SQLAlchemy, SQLObject, Storm (software), Tkinter, Topsite Templating System, Twisted (software), VPython, WxPython, XDMF, Pipenv, PyTorch, Caffe2, Pendulum, Dash, PyFlux, Fire, imbalanced-learn, FlashText, Luminoth, Scrapy, Pillow, NumPy, SciPy, matplotlib, Scapy, pywin32, nltk, nose, SymPy, IPython etc.

Open source library uses some open source software like Linux, Apache Web Server, OpenOffice, GIMP, Audacity, and Firefox browsers. Koha and Evergreen are referred to as integrated library systems (ILS). They are the most popular libraries and both are licensed under a GNU General Public license. Opens source application depends on open source libraries to perform certain operation or features. Example of the dependent open source applications are as follows: OpenCog, OpenCV, TREX, ROS, YARP, FreeCAD, BIM, LibreCAD, Blender, flightgear, SimPy, Scribus, Bitcoin Core, Bonita Open Solution, CiviCRM, Compiere, Cyclos, Dolibarr, ERPNext, GnuCash, HomeBank, iDempiere, Ino erp, jFin, JFire, KMyMoney, LedgerSMB, metasfresh, Mifos, Odoo, Openbravo, OrangeHRM, Postbooks, QuickFIX, QuickFIX/J, SQL Ledger, SugarCRM, Tryton, TurboCASH, Wave Accounting, ZipBooks, Evergreen, Koha, NewGenLib, OpenBiblio, PMB, refbase, Darktable, digiKam, GIMP, Inkscape, Krita, LightZone, RawTherapee, Chemistry Development Kit, JOELib, OpenBabel, P-GRADE Portal, CellProfiler, Endrov, FIJI (software), Ilastik, ImageJ, IMOD, ITK, KNIME, VTK, 3DSlicer, Abalone, Ascalaph Designer, GROMACS, LAMMPS, MDynaMix, NAMD, NWChem, Avogadro, BALLView, Jmol, Molekel, MeshLab, PyMOL, QuteMol, RasMol, Ninithi, CP2 K, LimeSurvey, CMU Sphinx, Emacspeak, ESpeak, Festival Speech Synthesis System, Modular Audio Recognition Framework, NonVisual Desktop Access, Text2Speech, Dasher, Gnopernicus Virtual Magnifying GlassEnvironment, Konstanz Information Miner (KNIME) OpenNN, Orange, RapidMiner, Scriptella ETL, Weka, JasperSoft, ParaView, VTK, ResourceSpace, ApexKB, Lucene, Nutch, Solr, Xapian, Elasticsearch, Konstanz Information Miner (KNIME), Pentaho, SpagoBI, Talend, OpenAFS, Tahoe-LAFS, CephFS, OpenX, Asterisk. Ekiga, FreePBX, FreeSWITCH, Jitsi, QuteCom, Enterprise Communications System sipXecs, Twinkle, Ring, Tox, Geary, Mozilla Thunderbird, GNU Queue, HTCondor OpenLava, pexec, Apache Axis2, Apache Geronimo, Bonita Open Solution, GlassFish, Jakarta Tomcat, JBoss Application Server, ObjectWeb JOnAS, TAO, Enduro/X, Akregator, Liferea, RSS Bandit, RSSOwl, Sage, Popcorn Time, qBittorrent, Drupal, Liferay, Oxwall, Sun Java System Portal Server, uPortal, FreeNX, OpenVPN, rdesktop, Synergy, VNC, Remmina, Brave, Chromium, Firefox, Midori, Tor Browser, Waterfox, SeaMonkey, Cheese, Guvcview,

cURL, HTTrack, Wget, Apache Cocoon Apache, AWStats, BookmarkSync, Cherokee, curl-loader, FileZilla, Hiawatha, HTTP File Server, lighttpd, Lucee, Nginx, NetKernel, Qcodo, Squid, Vaadin, Varnish, XAMPP, Zope, ATutor, Chamilo, Claroline, DoceboLMS, eFront, FlightPath, GCompris, Gnaural, H5P, IUP Portfolio, ILIAS, Moodle, OLAT, Omeka, openSIS, Sakai Project, SWAD, Tux Paint, UberStudent, KGeography, Kiten KVerbos WINE, CyberBrau, Pencil2D, Pivot Animator, Synfig, Tupi (formerly KTooN), OpenToonz, Blender, OpenFX, Seamless3d, Pencil2D, SWFTools, Eye of GNOME, F-spot, Geeqie, Gthumb, Gwenview, Kphotoalbum, Opticks, Dr. DivX, FFmpeg, MEncoder, OggConvert, Avidemux, AviSynth, Blender, Cinelerra, DScaler, DVD Flick, Flowblade, Kaltura, Kdenlive, Kino, LiVES, Natron, OpenShot Video Editor, Pitivi, Shotcut, VirtualDub, VirtualDubMod, VideoLAN Movie Creator, Avidemux, HandBrake, FFmpeg, Apache OpenOffice, Calligra Suite, LibreOffice, Chandler, KAddressBook, Kontact, KOrganizer, Mozilla Calendar, Novell Evolution, OpenSync, Project.net, TeamLab, Bugzilla, Mantis, Mindquarry, Redmine, Trac, Bison, CodeSynthesis XSD, CodeSynthesis XSD/e, Flex lexical analyser, Kodos, Open Scene Graph, OpenSCDP, phpCodeGenie, SableCC, SWIG xmlbeansxx, YAKINDU Statechart Tools, Doxygen, Mkd, Natural Docs, Autoconf, Automake, BuildAMation, CMake, GNU Debugger, Memtest86, Xnee, BOINC, Electric Sheep, XScreenSaver, MyDLP, dvdisaster, Foremost, PhotoRec, TestDisk, Mydiamo, Coyote Linux, Firestarter, IPFilter, ipfw, iptables, M0n0wall, PeerGuardian, PF, pfSense, Rope, Shorewall, SmoothWall, Untangle, Vyatta, Java Astrodynamics Toolkit (GPL), General Mission Analysis Tool (NASA Open Source Agreement), OREKIT (ORbits Extrapolation KIT) (Apache License), Satellite tracking and orbit prediction (GPL), Orbit Reconstruction Simulation and Analysis (GPL), Asteroid Orbit Determination and Propagation (GPL), Libnova (LGPL), Open-Source, Extensible Spacecraft Simulation And Modeling Environment (GPL), Distributed Spacecraft Attitude Control System Simulator (GPL), Solar Sail structure and flight simulator (GPL), SaVi satellite constellation visualizer (BSD License), Rocket Workbench Project (GPL), CpropepShell. Compute propellant performance, BRL-CAD, Blender, Blender CAD, Procad, OpenSCAD, Python CAD, VARKON, OpenCASCADE, FreeCAD, Archimedes, Wikipedia Free CAD Software Listing, Linux.org CAD/CAM Software Listing, NASA Vision Workbench (NOSA license), wikiCalc (GPL), Dia (GPL), DUNE (GPL with runtime exception), Impact (GPL), Code_Aster, SALOME (LGPL), Elmer, Gmsh, OpenFVM, Palabos, Calculix, Package of Additional Octave Libraries, ASCEND modelling environment, OpenDX (IBM), Freshmeat.net Visualization Software Listing, VisIt (BSD), EngLab (GPL), SciLab (CeCILL license), WorldWind (NOSA), Numpy/Scipy, OpenCog, AForge.NET, TREX, ROS, YARP, Blender, flightgear, Chemistry Development Kit, JOELib, CellProfiler, Endrov, FIJI, Ilastik, ImageJ, IMOD ITK, KNIME, VTK, 3DSlicer, Abalone, Ascalaph Designer, GROMACS, LAMMPS, MDynaMix, NAMD, NWChem, Avogadro, BALLView, Jmol, Molekel, MeshLab, PyMOL, QuteMol, RasMol, Ninithi, CP2 K, CMU Sphinx, Emacspeak, ESpeak, Bullet, AwayPhysics, Bullet-ANE, ammo.js, Physijs, AmmoNext, Bullet.js, JBullet, ODE, Bounce, nphysics, Velocity Raptor, Box2D, Nape, GoblinPhysics, verlet-js, PhysicsJS, Matter.js, p2.js, Coffee Physics, JPE, APE, Chipmunk2D, glaze, ImpulseEngine,

JelloAS3, JelloHx, Jello-Physics, JelloSwift, JigLib, Moby, Newton-Dynamics, OimoPhysics, qu3e, Tokamak, DynaMo, ReactPhysics3D, Chrono Engine, Position Based Dynamics, SPlisHSPlasH etc.

Modeler sometime desire to create his/her own libraries. The step is quite easy if the preliminary steps are diligently executed. For example, there must be an interface to the proposed library. The header file to the proposed library should contain definitions for everything exported by your library. This includes: function prototypes with comments for users of your library functions; definitions for types and global variables exported by your library. The modeler is expected to write a "boiler plate" code that enables the preprocessor to include the 'proposedlib.h' file one time. Aside creating the interface, the modeler is expected to design the implementation flow chart of the library. This exercise is achieved by creating a proposedlib.c file that #includes "proposedlib.h". The next step after creating implementation code is creating a library object file that can be linked with programs that can access the proposed library code. Alternatively, modeler may wish to create a shared object file from many.o files that can be linked with programs that want to use the proposed library code. However, before you share file, be sure of the license compatibility issues. The file sharing can be achieved by linking the proposed.c file with the library object file. An example of the.c file linking in C language is presented below.

" gcc test.c mylib.o
OR to link in libmylib.so (or libmylib.a):
gcc test.c -lmylib
OR to link with a library not in the standard path:
gcc test.c -L/home/newhall/lib –lmylib"

An example of the.c file linking in C ++ language is presented below.

"INC = -I ./Headers
g++ main.o proposedlib.o
g++ $(INC) –c main.cpp
g++ $(INC) –c proposedlib.cpp
rm proposedlib.o main.o a.out
make
./executable"

An example of linking your created module to the main module in python language is presented below.

```
“import proposedlib
if __name__ == "__main__":
import sys
fib(int(sys.argv[1]))”
```

An example for linking the C++ to the Perl program is extracted from Perlxs (2018). The produced Perl function will accept that its first contention is an object pointer. The object pointer will be put away in a variable called THIS. The object are written in C++ with the new( ) work and are executed by Perl with the sv_setref_pv( ) macro. The object by Perl can be dealt with by a typemap. For example, if the C++ code shown below

```
“    class colour {
      public:
       colour();
            ~colour();
       int blue();
       void set_blue( int );
      private:
      int c_blue;
  };
”
```

is to be linked to the Perl program using THIS.

```
“ RETVAL = THIS->blue();
THIS->set_blue( val );”
```

So that the link will look like below

```
"
  int
colour::blue( val = NO_INIT )
   int val
   PROTOTYPE $;$
 CODE:
     if (items > 1)
           THIS->set_blue( val );
       RETVAL = THIS->blue();
    OUTPUT:
       RETVAL
"
```

References

Capterra. (2017). *Environmental software*. https://www.capterra.com/environmental-software/. Accessed on January 4th, 2018.

Chandra, K. B., & Chinmayee, R. (2012). Incorporating hidden Markov model into anomaly detection technique for network intrusion detection. *International Journal of Computer Applications, 53*(11), 42–47.

ConnectUS (2018). 7 Main Advantages and Disadvantages of Open Source Software, https://connectusfund.org/7-main-advantages-and-disadvantages-of-open-source-software. Accessed November 12th, 2018.

Emetere, M. E., & Sanni, E. S. (2015). A Review on the comparative roles of mathematical softwares. *Global Journal of Pure and Applied Mathematics, 11*(6), 4937–4948.

Flory, W. J. (2018). *3 tips for organizing your open source project's workflow on GitHub*. https://opensource.com/article/18/4/keep-your-project-organized-git-repo. Accessed August 25th, 2018.

Higgs, P. (2016). *The disadvantages of open source*. https://www.gaiaresources.com.au/open-source/. Accessed January 4th, 2018.

Ibrahim, H. (2010). https://www.linuxfoundation.org/resources/open-source-guides/starting-open-source-project/. Accessed August 25th, 2018.

Igor, B. (2017). *Top 15 Python libraries for data science in 2017*. https://medium.com/activewizards-machine-learning-company/top-15-python-libraries-for-data-science-in-in-2017-ab61b4f9b4a7. Accessed on January 4th, 2018.

István, F., Ágnes, H., Zahari, Z. (2013). *Advanced numerical methods for complex environmental models: Needs and availability* (p. 201). https://doi.org/10.2174/97816080577881130101; eISBN: 978-1-60805-778-8, 2013. ISBN: 978-1-60805-777-1.

Korolkovas, A. (2016). *Entangled polymer flows at interfaces*. A thesis submitted to University of Upsala.

Matthias, S. (2013). *Four types of open source communities*. https://opensource.com/business/13/6/four-types-organizational-structures-within-open-source-communities. Accessed August 25th, 2018.

NASA. (2018). *Open-source software project*. https://code.nasa.gov/. Accessed January 4th, 2018.

NOAA. (2018). *Numerical weather prediction*. https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/numerical-weather-prediction. Accessed January 9th, 2018.

OSI. (2018). https://opensource.org. Accessed August 25th, 2018.

Patwardhan, M. (2016). *Assessing the impact of usability design features of an mHealth app on clinical protocol compliance using a mixed methods approach*. Arizona State University: ProQuest Dissertations Publishing; 2016. https://repository.asu.edu/attachments/172769/content/Patwardhan_asu_0010N_16210.pdf

Perlxs, (2018). http://perldoc.perl.org/perlxs.html#Using-XS-With-C%2b%2b. Accessed August 27th, 2018.

Seher, R. (2017). *Combining open source software licenses—The final chapter*. http://blog.thehyve.nl/blog/open-source-software-licenses-3. Accessed August 25th, 2018.

Tom, A. (2004). *How to misunderstand open source software development*. http://www.consultingtimes.com/ossdev.html. Accessed August 25th, 2018.

Upasani, O.S. (2016). Advantages and limitations of open source software for library management system functions: The experience of libraries in India. *The Serials Librarian, 71*(2), 121–130. https://doi.org/10.1080/0361526X.2016.1201786.

Välimäki, M. (2005). *The rise of open source licensing: A challenge to the use of intellectual property in the software industry* (*Ph.D. thesis*). Helsinki University of Technology. Retrieved 2015-12-30.

Wilbanks, J. (2013). *Understanding open science*. http://fastercures.tumblr.com/post/56790751132/understanding-open-science. Accessed August 24th, 2018.